Atomic Energy Control Board
P.O. Box 1046
Ottawa, Canada
K1P 5S9

Commission de contrôle de l'énergie atomique
C.P. 1046
Ottawa, Canada
K1P 5S9

INFO-0224

CONVERSION OF CATHENA FROM THE VAX TO APOLLO COMPUTER

by

D.M. Kawa
Wardrop Engineering Consultants

A research report prepared for the
Atomic Energy Control Board
Ottawa, Canada

AECB Project #86.1.22

December 1986

Canada

Research report

CONVERSION OF CATHENA FROM THE VAX TO APOLLO COMPUTER

ABSTRACT

The thermal-hydraulic computer code developed by Atomic Energy of Canada Limited, CATHENA, has been converted from the VAX computer to operate on a low-cost, high-performance APOLLO 3000 work station. Full functionality of the program has been retained.

Three verification cases have been analyzed which indicate accuracy to the sixth decimal place as printed in output listings.

The APOLLO 3000 computing times for the 3 cases are approximately 15% faster than achieved on the Wardrop VAX 11 750.

RÉSUMÉ

Le code machine thermo-hydraulique mis au point par l'Énergie Atomique du Canada, Limitée, CATHENA, a été converti à partir de l'ordinateur VAX pour fonctionner sur un poste de travail APOLLO 3000 à faible coût et à haut rendement, sans porter atteinte au fonctionnement intégral du programme.

On a procédé à trois analyses de vérifications de cas qui ont indiqué des résultats exacts à la sixième décimale près selon les listages imprimés.

La durée de calcul de l'APOLLO pour les trois cas a été d'environ 15 pour 100 plus courte que celle du VAX 11 750 de Wardrop.

DISCLAIMER

The Atomic Energy Control Board is not responsible for the accuracy of the statements made or opinions expressed in this publication and neither the Board nor the author assumes liability with respect to any damage or loss incurred as a result of the use made of the information contained in this publication.

TABLE OF CONTENTS

ABSTRACT..... i
A. INTRODUCTION..... 1
1. Description of CATHENA..... 1
2. Purpose of Conversion..... 1
3. Scope of Conversion..... 1
4. Contract Identification..... 2
B. OVERVIEW OF CONVERSION..... 3
1. Method of Conversion..... 3
2. Files Requiring Modification..... 3
3. APOLLO and VAX File Name Differences..... 4
4. Command Procedure Files..... 4
C. CONVERSION MECHANICS..... 5
1. Description of The CVA Program..... 5
2. Description of Modifications Performed by CVA..... 5
3. Transfer of Output from CVA to APOLLO..... 7
D. CONVERSION OF SYSTEM DEPENDENT ROUTINES..... 8
1. System Dependant Routine Modifications..... 8
E. COMPILER DIFFERENCES..... 11
F. CODING BUGS..... 16
G. FORMAT STATEMENTS..... 17
H. FILE ASSIGNMENT, AND OPENING OR CLOSING OF FILES..... 18
I. PLOTTING..... 20
1. Generation of Solution Data for Plotting..... 20
2. Postprocessing using the PLOT or PRINT Option..... 20
J. VERIFICATION..... 23
1. Verification Cases..... 23
2. Modifications to Input Files..... 23
3. Comparison of Output From APOLLO and VAX..... 25
4. RD14 Input File..... 25
K. VAX AND APOLLO COMPUTING TIMES..... 26

L. OPERATION OF CATHENA ON THE APOLLO..... 27
1. Command Procedures..... 27
2. Restart Capability..... 27
3. Plotting..... 28
4. Printing..... 28
M. KNOWN PROBLEMS IN THE APOLLO SOURCE CODE..... 29
N. RECOMMENDATIONS..... 30
TABLE 1..... 31

APPENDIX A CVA (CONVERT VAX TO APOLLO) SOURCE CODE LISTING.
APPENDIX B TRANSFER OF FILES FROM VAX TO APOLLO
APPENDIX C SAMPLE TEKTRONIX INTERACTIVE PLOTS

A. INTRODUCTION

1. Description Of CATHENA

The CATHENA program is being developed by Atomic Energy of Canada Ltd. (AECL) at the Whiteshell Nuclear Research Establishment (WNRE) Research Company. The program is used for computation of coupled nuclear reactor thermal hydraulic/heat transfer behaviour of nuclear systems for analyzing a variety of operating and accident conditions.

2. Purpose Of Conversion

CATHENA currently runs on DEC VAX and CDC computers. The price performance ratio of these computers is relatively high in comparison to the latest generation of hardware. It is therefore desireable to have CATHENA available on lower priced high performance computers such as the APOLLO 3000 monochrome work station to reduce computer costs for analyzing reactor incidents. This report discusses the details of this project undertaken by Wardrop Engineering Consultants for the Atomic Energy Control Board (AECB) for converting the VAX version of CATHENA to the APOLLO 3000 workstation.

3. Scope Of Conversion

The scope of work included obtaining the source code from AECL, converting to the APOLLO, verifying the conversion through running sample cases, and documenting the results.

4. Contract Identification

This work was produced by Wardrop Engineering Consultants under contract to the Atomic Energy Control Board; contract no. 87055-6-4087/01-SS.

B. OVERVIEW OF CONVERSION

1. Method Of Conversion

The conversion was performed by altering the VAX source code to minimize the machine dependence between the VAX and APOLLO. This code was maintained concurrently on the two machines. A run would be attempted on the APOLLO, during which errors would occur. The VAX would then be used to locate and correct the problemand search the remainder of the code for similar errors. The source code would be modified on the VAX, transferred to the APOLLO, and the process repeated. A series of approximately 30 iterations was required to convert the code to the APOLLO.

2. Files Requiring Modification

The resulting source code can be run on either the APOLLO or VAX with a minimum number of changes. The following files are those with APOLLO or VAX dependency. The first 3 are general CATHENA modifications while the remainder in the list have plot dependant routines.

File	Changes
CATHENA	- CPUTYP is "VAX" or "APOLLO".
CPTIME	- Use general VAX or APOLLO library functions.
DAYTIM	- Use general VAX or APOLLO library functions.
GPRPLOT	- General Tektronix or APOLLO plot changes.
GPRPLOTS	- General Tektronix or APOLLO plot changes.
GPRSYMBOL	- General Tektronix or APOLLO plot changes.
PLOTS	- CLOSE statements (3 locations) APOLLO or VAX.
BLOCKDATA	- APOLLO or VAX Hexadecimal values for "SOLID" and "DOTTED".

Table 1 summarizes all the changes made to each file in the conversion of the VAX source code to the APOLLO. Most changes listed result in a source code which, after being processed by CVA, is transparent to both the APOLLO and VAX, except for the routines listed above.

3. APOLLO And VAX File Name Differences

The naming of files between the 2 computers differs. The Fortran source files on the APOLLO are given a ".FTN" extension while the VAX files have a ".FOR" extension. In addition to the Fortran files, a series of common files are used by the various modules, incorporated through the Fortran "INCLUDE" directive. The common files are given the extension ".CMN" on both machines.

4. Command Procedure Files

Three command procedure files have been included with the names FTNOPT.COM, FTN.COM, and BIND.COM. These files have the APOLLO commands for compiling the source code with or without optimization, and for binding the binary modules produced by the Fortran compiler into an executable program. The compiler switches used in FTN.COM are -COND, -NPEB, -SAVE, -ZERO, and -INDEXL. The file FTNOPT.COM uses the switches -COND, -CPU 3000, -OPT, -SAVE, -ZERO, and -INDEXL.

The APOLLO commands for compiling all routines are $FTN.COM or $FTNOPT.COM and the binding command is $BIND *BIND.COM.

C. CONVERSION MECHANICS

1. Description Of The CVA Program

The CATHENA program consists of more than 500 subroutines, with approximately 50,000 lines of Fortran source code. Repetitive checking of source lines was required in the conversion. To aid in conversion, a Fortran program CVA (Convert Vax to Apollo) was developed to perform the bulk of conversion. This program is listed in Appendix A.

Many iterations were required in the conversion. The source code would be transferred to the APOLLO, a run made, an error encountered, and a change would be incorporated into CVA or the VAX CATHENA source file. The source files would be regenerated and the process repeated.

2. Description Of Modifications Performed By CVA

The following list summarizes the final items addressed in the conversion program CVA.

(a) All VAX file extensions '.FOR' were renamed to '.FTN' on the APOLLO. The INCLUDE file extensions '.CMN' were retained for the APOLLO version.

(b) Comment statements were transferred directly to the APOLLO with no modification.

(c) The 'INCLUDE' statements in VAX CATHENA reference a logical directory name as part of the file specification which was used to reference development directories for the associated files. These are not required in the APOLLO version, and have been edited out of the file specification in each INCLUDE

statement. The logic used for removing these references was to search for an INCLUDE statement, delete the text following the first single quote (') to and including the characters : which terminate the VAX logical directory name.

The appearance of the resulting statement which replaced the VAX statement follows:

```
%INCLUDE 'filename.CMN'
```

with the % located in column 1.

(d) The APOLLO does not permit TAB characters in the source code. The conversion program changes each TAB to 6 blanks or terminates the TAB at column 6 in the source statement so that the Fortran text starts in column 7. An exception exists for continuation lines where the digit character following the TAB is placed in column 6. Each statement, except for comments, was checked for continuation on the next line if modifications exceed column 72. The continuation logic was included in the CVA program.

(e) The VAX "ACCEPT *," statements were changed to "READ(5,*)" statements.

(f) VAX in-line comments "!" were changed to "{".

Additional editing of the CATHENA source files was performed to make minor changes which were not feasible within CVA. These additional changes are discussed in this report.

3. Transfer Of Output From CVA To APOLLO

Each CATHENA source file on the Wardrop VAX was processed by CVA and concatenated into one large master file. Each individual source file was labelled with a special header record containing the file name. All files were transferred to the APOLLO by magnetic tape using the APOLLO RWMT (Read Write Mag Tape) program, or through a serial input/output (IO) communications port using the APOLLO EMT module (EMulate Terminal) for capturing the transmitted data and writing the information to a disk file. A Fortran program, SPLIT, was transferred to the Apollo for splitting the large file into each individual file by keying on the header record for each new file.

The transfer of files from the VAX to the APOLLO using EMT and RWMT are discussed in Appendix B.

D. CONVERSION OF SYSTEM DEPENDENT ROUTINES

WNRE has developed 2 versions of CATHENA. One is the VAX version which is most compatible to the APOLLO. The second is the CDC version. System dependant source routines were selected from the machine providing greatest compatibility with the APOLLO. For example, naming the system in SETSYS was chosen from the CDC version, while word structure routines were chosen from the VAX version.

1. System Dependant Routine Modifications

The following list discusses the general system dependant subroutines used in the conversion.

ALLOC - This routine was used for allocating memory dynamically. Memory on the VAX is controlled by allocating a block of memory at run time having an offset from the blank common matrix JJA(1). The APOLLO does not support allocation of memory during program execution, and therefore, this logic cannot be used. Instead, a blank common matrix JJA was dimensioned to a large size in ALLOC. This array is made available to other subroutines through blank common referenced through the common files DYNMAT.CMN, DYNMOD.CMN, DYNIOT.CMN, and DYNWRK.CMN, and a pointer to the offset storage location. For solving larger problems, the PARAMETER variable NWJJA in subroutine ALLOC will have to be modified, the routine recompiled, and all binaries bound again.

CPTIME - Subroutine CPTIME returns the CPU time used by the current process. The APOLLO calls:

```
CALL PROC1$GETCPUT(CLOCK)
CALL CAL$FLOATCLOCK(CLOCK,SECONDS)
```

were implemented to return the CPU time in SECONDS.

DAYTIM - Subroutine DAYTIM is used to return date and time of day. The APOLLO calls:

```
CALL CAL$GETLOCALTIME(CLOCK)          !APOLLO
CALL CAL$DECODETIME(CLOCK,DATE)       !APOLLO
```

were made to return the date and time in array DATE. The date integers DATE were converted to the VAX ASCII date format be ENCODing the DATE values into array V.

SETSYS - the computer system name was set to 'AP3000' for the Apollo 3000 work station.

STPRAM - A CATHENA run can be stopped using STPRAM. This routine was used on the VAX for checking parameters such as remaining disk space, batch CPU time, etc, and setting a flag for stopping execution as soon as possible. (The CDC call to XTIME, remaining batch CPU time was bypassed.) The APOLLO routine only checks for the solution time reached, for the existence of a file named STOP.RAM, or a trip, and set a flag for stopping so that a restart record can be written.

Linking (Binding) the original VAX routines at Wardrop resulted in 5 unresolved routine calls (CHANGE, DCLOSE2, IDENT, PWRITE2, and VRINIT). These are the Versatec plotter subroutines which are not required for the APOLLO and were commented out in subroutine PLOTS.

E. COMPILER DIFFERENCES

A series of general changes were made which related to basic differences between the two compilers. The following list presents all general changes made within CATHENA to permit operation on the APOLLO computer. These changes, in general, standardize CATHENA to eliminate future compiler restrictions. The modifications are listed alphabetically by subroutine with a short description of the modification required. The subroutine names are capitalized and appear first in the description. Multiple routines having the same error are listed together.

(a) ANULF had two 'INCLUDE' statements containing the VAX '/LIST' directive. The /LIST option was removed using the VAX system editer. If the LIST is desired on the APOLLO, a %LIST statement should precede the %INCLUDE statement, and a %NOLIST statement should follow. The % character is to be located in column 1 of the statement.

(b) AXEL, AXIS and CODL made calls to COSD and SIND. The calls to COSD and SIND were modified by dividing each argument by 57.29578 to convert from degrees to radians, and making the calls to COS and SIN respectively.

(c) AXES and CODL had a LOGICAL*1 statement which the APOLLO does not accept. These were changed to LOGICAL. In AXES, the CHARACTER Z1*12 was changed to Z1*13, and an extra comma was added to the FORMAT statement at label 70, changing 'I' to ',I'.

(d) BLINDEA, BLINDEB, BLINDEC, BLINDED, and NCPROP encounter RETURN statements prior to executing any code. This was done intentionally by AECL. A compiler warning is generated for each of these routines.

(e) CATHENA and IHSLD were modified so that the first comment statement was placed after the routine name to prevent generation of a new file header name by CVA.

(f) CIRC, PIEC, and PLOTS had VAX TYPE statements for output to the VAX screen in interactive sessions. The APOLLO does not permit TYPE statements, and therefore, these have been converted to PRINT statements using the VAX system editer.

(g) CODL made a call to the plotter routine NEWPEN. This call is not required and was commented out. In addition, CODL had a statement with an extraneous brackets which had to be removed. Statement

WRITE(24) ((XX(I),YY(I)),I=1,NUM) was changed to

WRITE(24) (XX(I),YY(I) ,I=1,NUM) on the VAX since

the APOLLO would not accept the unnecessary brackets. In addition, the VAX logic .EQ. for variable IRT at the end of the routine was changed to incorporate the CDC logic since the APOLLO would not permit the mixing of LOGICAL and INTEGER computations.

(h) CUROT was modified to pass double precision constants in the CALL CUROOT argument list, 2 places.

(i) FLUXBUG, HITCHFCK, HITDBPRI, HITDEBUG, INWLBUG, WALLPTS, WALLPTT, and WALLPTU used Fortran Dlines on the INCLUDE statements. The APOLLO does not handle a Dline on an INCLUDE statement since the %INCLUDE must appear in column 1. Therefore, the 'D' in column 1 was removed on the corresponding INCLUDE statements. The ".CMN" files referenced on the Dlines were FLXBUG, HITBUG, WLINBUG, and WALLBUG. These ".CMN" files were modified to contain the Dlines, thereby achieving the same function. To eliminate unresolved global references on the Dline subroutines, and to be compatible with the removal of the Dlines on the INCLUDE statements in these subroutines, the Dlines were removed from the SUBROUTINE, RETURN, and END statements in each of the listed subroutines. Because the Fortran source files contain Dlines for conditional compiles, the APOLLO compile switch "-COND" was set for each subroutine to invoke the Dlines conditional compile.

(j) HEATFLX and HTWALLF had undefined integers IBUGHEA and IBUGWAL referenced as logical variables. These variables are used for controlling debug printout. Each was declared LOGICAL and set to .FALSE..

(k) HEATFLX was modified to pass double precision constants to PRESTR, 2 places.

(l) HTBCURV was modified by adding the statement "EXTERNAL CONVECT" since "CONVECT" is passed to another subroutine in an EXTERNAL context.

(m) INPMOD1, TDVMOD2, and STPRAM compared Hollerith variables to Hollerith constants of different word length which the APOLLO does not permit. Therefore, where these comparisons are performed, the Hollerith constant is placed in a variable of the same name preceded by an "H", eg, 4HTIME is converted to "HTIME". The routines where the Holerith comparisons have been altered, including the new Holerith variables are:

INPMOD1 - HTIME
TDVMOD2 - HCONTROL, HRCMODEL
STPRAM - BLANKS

(n) INPTDVA argument list variable CTAGI was not declared character, and therefore produced an error on the call to TRCHAR through the addressing.

(o) INTBICA and INTBICB were modified to ensure the CALL INTEG argument (1.Ø-RF) passed to INTEG would be double precision. The change incorporated was (1.ØDØ-RF). INTBICA had 3 calls to INTEG, and INTBICB had 6 calls.

(p) INTBICD was modified because of two statements of the form:

```
CALL HTDBPRI( 'INTBICD-' // FLOWREG , 'LOCAL' )
```

The APOLLO does not permit character concatenation in subroutine argument lists. These statements were modified as follows:

```
CHARACTER*3Ø ICHARCONCATE
ICHARCONCATE = 'INTBICD-'//FLOWREG
CALL HTDBPRI( ICHARCONCATE , 'LOCAL' )
```

(q) INWLBUG compared the PARAMETER variables IROUIN and IROUEXE to 40 in IF statements. The APOLLO does not permit comparison of PARAMETER variables to constants. A variable IJK was set equal to these parameter variables, with the comparison made to IJK.

(r) MOVEI was modified by commenting the ENTRY MOVEW statement. A new subroutine MOVEW was added to permit easy location and access to the file. The function was duplicated.

(s) MOVES, RSTFND and RSTINP were modified by changing the REAL statement to REAL*4, to eliminate any possibility of the IMPLICIT REAL*8 redefining the real variables in the REAL statement from 4 bytes to 8 bytes.

(t) MOVEW was added to perform the same function as the ENTRY MOVEW statement in subroutine MOVEI.

(u) PRNTLNK generated a compiler warning for the DATA statement defining TYPEC. The size initialized was 16 for a declared dimension of 17. One extra blank word was initialized at the end of the array in the data statement.

(v) PUTSP had a SQRT(-1.ØDØ) statement for producing a fatal runtime error. The APOLLO compiler would not accept this statement. This was replaced as follows:

```
ZNEG=-1.ØDØ
SQRT(ZNEG)
```

(w) RSTINP2 was modified to include a call to a new subroutine RSTINP8 used to convert single precision real variables in the restart file to double precision. (See RSTINP8 for a discussion of word structure differences on the APOLLO which affect the single/double precision argument list passing mechanism very significantly.

(x) RSTINP8 is a new subroutine used to convert single precision real (32 bit) variables in the restart file to double precision (64 bit) variables. The REAL word structure differs between the VAX and APOLLO. The VAX stores exponents in the low 32 bits, and the added precision in the high 32 bits. Therefore, reading the restart file directly into "V(1,I)" of the array dimensioned as "V(2,*)" simply truncates the high 32 bits in RSTINP2, defining the real variable in the low 32 bits. The APOLLO differs in that the exponent is stored in the high 32 bits, thereby requiring new logic to transfer the exponent into the high 32 bits. RSTINP8 was added to transfer to the double precision array correctly for both machines.

(y) SLVSYS1 had the DLINES changed to CDLINES since these lines are NOT to be included in the conditional compile, and therefore exist in the source as reference comment statements only.

(z) SPLITH had a Dline missing on the WRITE statement located 6 lines before the END statement. The Dline was added.

(aa) SUMEQ was modified to pass double precision constants to HTLOCAL (4 places in two calls), and HTWALL one place.

(bb) SURGER1 had a character specification statement "CHARACTER*16, ". The comma in the statement was removed.

(cc) TPLOT calls the plotting subroutines which require single precision arguments for the real variables. Therefore, XOFF was declared REAL*4 for argument compatibility on the APOLLO.

F. CODING BUGS

Several code bugs were located which had the potential for producing erroneous results in a run. These bugs are listed as follows:

(a) INWL2HA at DO loop 1003 could have potentially overwritten memory outside the defined size of array NUMCHA(8) if numeric names were supplied in the input data. In addition, the array NUMCHA() requires initializing within loop 1003 if more than one call is made to this routine, and the character data is numeric.

(b) INWL4B at DO loop 1005 could have potentially overwritten memory outside the defined size of array NUMCHA(8). See the discussion on INWL2HA.

(c) TRIM is used to delete leading and trailing blanks in character strings for the plotting function. An error occurs since TRIM assumes that every string is 80 characters long whereas at least one situation exists where the string is only 40 characters in length. This results in a comparison of undefined data from positions 41 to 80. Coding was added to use the LEN function to determine the string length prior to detection of non-blank entries.

(d) OPENF (VAX version) references variable "ACCESS(3)" rather than ACCESS(2), for append mode, 2 places. This appears to be an error on the VAX. The APOLLO version was modified appropriately.

G. FORMAT STATEMENTS

The APOLLO is more restrictive than the VAX regarding imbedded commas for separating entries in FORMAT statements. The following list summarizes the changes required for the APOLLO. In addition, obvious field width problems which exist in the WNRE VAX version were corrected and are reported.

(a) Several subroutines had format statements missing obvious commas which the VAX assumes, but the APOLLO does not. The routines and associated format statement numbers requiring the additional commas are:

ATHPRI — 460 FORMAT
AXES — 70 FORMAT
WALL2 — 975 FORMAT
WALL6A — 975 FORMAT
HTDBPRI— 979 FORMAT
HTDBPRI— 981 FORMAT
PRNTNC — 50 FORMAT 0PI6 was changed to 0P,I6

(b) INPCNTL had 5 print statments which exceeded format width, producing a series of '********' in the output. Eight variables were modified in print statements using format numbers 20, 30, 40, 90 and 100. The variables TSTART, TFIN, PRTHO, PRMDO, RSTO, TCNTLP, VMIN and PDBGT were changed to TSTARTDUM, TFINDUM, PRTHODUM, PRMDODUM, RSTODUM, TCNTLPDUM, VMINDUM and PDBGTDUM respectively. The 'DUM' variables were set to maximum format width if the format width would be exceeded.

(c) The plotting subroutines FINDMOD, FINDPOS, and GETDEF use FORMAT statements that are not recognized on the APOLLO, specifically, FORMAT(F) and FORMAT(I). These were modified to runtime FORMATs constructed by counting the number of characters to be read in the string.

H. FILE ASSIGNMENT, AND OPENING OR CLOSING OF FILES

The file OPEN statement of the two machines is quite similar. However, several differences do exist which required programming changes in the OPEN statement used in subroutine OPENF.

A difference also occurs in the external assignment of file names to Fortran logical unit numbers (luns). The VAX permits assignment prior to program initiation, while the APOLLO does not. Therefore, CATHENA was modified to OPEN the IO files at initiation of each run.

The following list summarizes the changes to the source code for internal file manipulation:

(a) OPENF is used for opening disk files. A common deck was incorporated into this subroutine for defining the machine type (APOLLO or VAX) so that the argument list in the OPEN statement could be constructed to be compatible with the appropriate machine.

(b) The APOLLO opens new disk files with a default maximum record length of 256 bytes. The records on the restart file are significantly larger and therefore, new restart files must be opened with a large buffer size. Subroutine INPIC calls OPENF for creating a new restart file. Two calls to OPENF were modified. The 'fix' used was to set the logical unit number to a negative value which OPENF then recognizes as the APOLLO restart file requiring a large buffer size. A large buffer size (RECL=8000 longwords) is set using the RECL keyword in the OPEN statement. To modify the buffer size, edit file OPENF, change the value of IRECL, compile OPENF, and BIND the program.

(c) Subroutines WATCH and INWLWAT open the log file with a VAX SHARED parameter so that users can monitor progress of batch jobs. The Apollo does not permit the SHARED parameter. It was therefore removed from these OPEN statements.

keyword by default.

(d) Three basic files are used in a run. These are the input file (FILE.INP), log file (FILE.LG), and output listing (FILE.LIS). Coding was added in the main program CATHENA to query the user for the FILEname to be used in the naming convention FILE.INP, and FILE.LG. These files are opened as Fortran logical unit numbers 5, and 2 respectively. Running the program involves preparation of the input file eg, SAMPLE.INP, a second file, say RUN containing the word SAMPLE, the prefix name of the input file. The run is intitiated by entering the APOLLO command:

```
$CATHENA   <RUN   >SAMPLE.LIS
```

The 2 files SAMPLE.INP and SAMPLE.LG, are opened in CATHENA using the prefix name specified in input file RUN, and SAMPLE.LIS is opened by the operating system to capture output data from the run.

(e) PLOTS has 3 CLOSE statements with a DISPOSE='DELETE' qualifier not recognized on the APOLLO. These were changed to a CLOSE statement with a STATUS='DELETE' argument.

I. PLOTTING

1. Generation Of Solution Data For Plotting

The CATHENA program has been structured to perform in a batch environment. A solution must be first generated using the RUN option for solving the transient or steady state problem. A restart file is generated which contains all computed variables written at the user specified time increment, or number of solution steps. The restart file can be subsequently accessed in a RUN mode to continue the transient and add data to the restart file in an APPEND mode, or in a post processing mode to extract computed variables for inspection using the PLOT or PRINT options.

2. Postprocessing Using The PLOT Or PRINT Option

The post processing modes available on the VAX are PRINT or PLOT. In either case, a new file (LUN 24) is written containing either the ASCII print columns of requested data, (eg, time in column 1, pressure at a specified node in column 2, etc), or the equivalent binary CALCOMP calls with argument data written as records in file VIDEO.PLT. Wardrop modified the plot routines so that the binary data being written to the LUN 24 file could be plotted interactively on a Tektronix screen using Tektronix calls, or APOLLO screen using the Graphics PRimatives (GPR) calls. The coding for both graphics packages is included in the 3 basic plot routines GPRPLOTS, GPRPLOT, and GPRSYMBOL. The Tektronix interactive plotting was developed on the VAX version, but interactive plotting was not completed for the APOLLO. Sample interactive Tetronix plots are given in Appendix C.

The PRINT option can be used by AECB for generation of ASCII columnar data which can be transferred to the Hewlett Packard plotter for generation of hard copies of the results.

Incorporation of the on-line screen graphics necessitated some changes and renaming of routines in the original CATHENA program. These changes are

listed.

(a) The VAX version of CATHENA used dummy routines which simply wrote the arguments to LUN 24, having no logic for plotting. These routines were AXEL, AXES, CODL, CWIDTH, PLOT, PLOTS, NUMBER, PLTCHK, and SYMBOL. These routines were replaced by the appropriate CALCOMP equivalent subroutines which reduced the graphics to three basic calls, initiation of plot, moving or plotting, and plotting symbols. The CATHENA routines PLOTS, PLOT, and SYMBOL have been modified to include calls to 3 new routines GPRPLOTS, GPRPLOT, and GPRSYMBOL which perform the necessary screen plotting to the APOLLO or Tektronix, depending upon which statements have been enabled in the source code.

Incorporation of the Tektronix library necessitated renaming the CATHENA routine "DRAW" to "DRAWP" and its associated calls from AXEL and AXES, in order to remove a conflict with the Tektronix library routine DRAW.

Certain of the plot routines are present in CATHENA, but have not been implemented by WNRE. These are ARCS, CIRC, and PIEC. Dummy calls have been set up to future routines GPRARCS, GPRCIRC, and GPRPIEC.

Plotting to the screen is performed automatically. The plot is held on the screen until a carriage return <CR> is depressed, whereby the screen is cleared and the next plot displayed.

(b) PLOTIN was modified to close the plot input specification file so that subsequent screen plotting could proceed with the keyboard <CR> used to control display duration. This file was modified to include the following CLOSE statement so that the keyboard can be used for control.

```
IF ( PLOTTYP.EQ.EOP ) THEN
  K1=0
  K2=0
CLOSE(UNIT=IN)              !<<< added to subroutine PLOTIN.
```

(c) SKPNXT is used to prevent the next plot instruction from being performed. Logic was inserted in subroutines SKPNXT, GPRPLOT, and GPRSYMBOL to bypass the plot instruction if SKPNXT was called.

(d) NUMBER is used to convert numeric data to characters for plotting. The routine as supplied to Wardrop would not operate properly and was rewritten by Wardrop.

(e) PLOTS was modified to incorporate an initialization call to GPRPLOTS.

(f) The interactive screen plotting file PLTCHK contained a dummy ENTRY AXISF statement which was commented out since AXISF exists as a dummy routine in the source files.

(g) No routine exists for NEWPEN. Therefore, PLTCHK was modified to include a dummy ENTRY NEWPEN statement.

(h) Unresolved global symbols were referenced in subroutine PLOTS. These were the Versatec plot routines CHANGE, DCLOSE2, IDENT, PWRITE2, and VRINIT. The statements with these calls were commented out.

(i) BLOCKDATA for plotting was modified. The data statement at the end of the routine referencing hexadecimal values for variables SOLID and DOTTED were changed to:

```
DATA SOLID /16 FFFFFFFF/, DOTTED /16 11111111/
```

for APOLLO hexadecimal representation.

J. VERIFICATION

1. Verification Cases

Three examples were supplied by WNRE to Wardrop for verification of the converted CATHENA source code. These sample cases, identified as EX1, EX2, and EX3, were developed by WNRE for exercising most of the source code.

2. Modifications To Input Files

EX1 made reference to an 'EMBECIL' heat transfer model which is not supported. This reference to EMBECIL was removed from the input file. All references to EX1 in this report will be to to non-EMBECIL model.

Note that the different examples are referred to as either EX1, EX2, EX3 or AX1, AX2, AX3 for depicting the examples as received from WNRE, or as modified on the Apollo.

File EX2.INP was edited and approximately 10 extraneous Carriage Returns <CR> were removed from the input file. In addition, EX2.INP had two list directed input lines which the APOLLO does not interpret correctly. The lines occurred in the 'TIMESTEP CONTROL' and 'JUNCTION RESISTANCE' definition as follows:

```
'TIMESTEP CONT'/
 , , , , , , , ,9.E5,9.E5/     (was originally 2*9.E5/)

'JUNCTION RESIST', 'JRES1'/
'R-RW 6 U','L-RW 7 D'/
1.D30,1.D30/                   (was originally 2*1.D30/)
```

EX3.INP had similar input difficulties with list directed input in seven

locations:

```
'TIME STEP CONTROL'/
1.0,.FALSE.,,,,,,,9.9E9,9.9E9/   (changed from 2*9.9E9/)

'JUNCTION RESISTANCE','D2OFVP3'/ MV13
'R-F1PUMP3','L-F2PUMP3'/
60.,60./                    (Changed from 2*60./)

'JUNCTION RESISTANCE','D2OFVP1'/ MV22
'R-F1PUMP1','L-F2PUMP1'/
60.,60./                    (Changed from 2*60./)


'INPUT TABLE','D2O-ISO'/ D2O ISOLATION VALVE CONTROL
4,6,,.TRUE./
'TIME','FKMV13','FKMV22','RKMV13','RKMV22'/
0000.00,60.,60.,60.,60./          (Changed from 0000.00,4*60./)
538.000,60.,60.,60.,60./          (Changed from 538.000,4*60./)
538.500,2.0E25,2.0E25,2.0E25,2.0E25/
612.000,2.0E25,2.0E25,2.0E25,2.0E25/
612.500,60.,60.,60.,60./          (Changed from 612.500,4*60./)
4494.00,60.,60.,60.,60./          (Changed from 4494.00,4*60./)
```

From cases EX2 and EX3, it appears that the APOLLO does not interpret the repeat feature "*" consistently in list directed input. The changes to the input files involved removal of the "*" repeat feature, entering the numbers individually.

3. Comparison Of Output From APOLLO And VAX

Following modification of the input files, the 3 examples were run on an APOLLO 3000 work station with and without source code optimization for floating point calculations. The optimized runs with the math co-processor chip produced slightly different results on EX2 in the 6th decimal place of certain printed numbers.

The three cases were run on the APOLLO, and the output was compared against the WNRE output files using the APOLLO utility CMF (CoMpare Files). The differences encountered in the output files (*.LIS) were the machine IDs, the dates, and occasional differences in the 6'th printed decimal place.

4. RD14 Input File

A fourth input file, SS8513.INP, which is an RD14 input file, was supplied for reference purposes. No computer runs were performed with this file.

K. VAX AND APOLLO COMPUTING TIMES

The following table presents the computing times on the APOLLO and VAX computers for each case. The conditions evaluated for the VAX were simply normal optimization of source code and hardware floating point accelerator FPA. APOLLO cases were run with no source code optimization and no math co-processor. A second case was run with full optimization and math co-processor. (The times are given in CPU minutes).

	VAX 8650 WC4	VAX 750 WLW1	APOLLO 3000 NPEB	APOLLO 3000 CPU3000 OPT
EX1	.91	6.0	7.74	4.96
EX2	10.9	76.9	106.3	69.3
EX3	7.95	59.2	74.0	47.4

L. OPERATION OF CATHENA ON THE APOLLO

1. Command Procedures

The APOLLO does not permit two files of the same name to exist concurrently in a directory. Therefore, a command procedure is recommended for running CATHENA. This procedure can be used for deleting or renaming files, defining the IO files, and running CATHENA. A sample procedure follows which deletes those files which will cause the run to terminate because of duplicate names in a directory.

```
DLF AX3.LG
DLF AX3.LIS
DLF WOL.SDY
CATHENA <AX3 >AX3.LIS
CHN WOL.SDY WOLL.SDY
```

In the procedure above, file AX3 is the input file which contains the input file name "AX3" with an assumed extension ".INP".

2. Restart Capability

A CATHENA run can be restarted after termination. The changes necessary to the input file are to specify TSTART in Solution Control, and the FILEIN file name on the Restart Control card. The revised input file, along with the restart file are accessed and the solution is propagated.

3. Plotting

Interactive plotting on the APOLLO screen is not available at this time, but is desirable for previewing results. The option is invoked by specifying the PLOT option on the Processing Option card. A plot definition file is also required.

4. Printing

Specifying PRINT on the Processing Option card can be used to generate an ASCII file of data columns for plotting on the HP plotter at AECB. A file name is specified to receive the user defined columnar data.

M. KNOWN PROBLEMS IN THE APOLLO SOURCE CODE

A potential problem may exist with regard to variables not being preset to zero before each CATHENA run. The uninitialized variables in CATHENA may assume non-zero values at the start of execution. Use of the -ZERO compiler option may not guarantee initialization to zero of uninitialized variables because the discussion in the DOMAIN Fortran User's Guide, Order No. 000530, Revision 04, Release 7.0, page 5-134, paragraph 2 suggests that a new shell must be initiated to preset memory to zero. It is therefore recommended that each CATHENA run be started from a new SHELL, since invoking a SHELL zeros the memory. (WNRE are attempting to define all uninitialized variables within CATHENA).

The second problem is associated with the APOLLO list directed input. Obvious errors occurred while running verification cases EX2 and EX3 in which real variables were specified in the input file using the list directed repeat feature, eg "2*60./". The APOLLO would not accept this type of input line consistently, although it does accept the repeat "*" feature on other input lines. The files and lines requiring modification are discussed under the heading VERIFICATION.

N. RECOMMENDATIONS

Wardrop proposes that implementation of the following recommendations will reduce model preparation time and interpretation of results very significantly.

(a) Develop a module for plotting the input geometry and network.

(b) Complete development of the interactive plotting of results. Wardrop attempted to develop this option on the APOLLO but could not complete the work within the contract time constraints, since this option was beyond the scope of the contract.

TABLE 1
Summary of CATHENA file modifications

"INCLUDE" .CMN files modified for the APOLLO -

CPUTYP - Added for definition of CPUTYP.
FLXBUG - Dlines added.
HTBUG - Dlines added.
WALLBUG - Dlines added.
WLINBUG - Dlines added.

FORTRAN files modified for the APOLLO -

ALLOC - Fixed dimensions in blank common, initializes new memory.
CPTIME - APOLLO specific system calls.
DAYTIM - APOLLO specific system calls.
OPENF - APOLLO compatible open statements, added CPUTYP.
SETSYS - Define the APOLLO machine type.
STPRAM - Routine for stopping transient run.

AXEL - Tektronix and APOLLO plots, renamed DRAW to DRAWP.
AXES - Tektronix and APOLLO plots, renamed DRAW to DRAWP, Z1, 70 FORMAT.
BLOCKDATA - Added for Tektronix and APOLLO plotting, hexadecimal values.
CODL - Tek and APOLLO plotting, CDC logic for dot-dash, added CPUTYP.
CWIDTH - Tektronix and APOLLO plotting.
DRAWP - Tektronix and APOLLO plots, renamed DRAW to DRAWP.
NOSD - Tektronix and APOLLO plotting.
NUMBER - Tektronix and APOLLO plotting.
PLTCHK - Tektronix and APOLLO plotting.

ANULF - INCLUDE "/LIST" option removed (2 places).
ARCS - Plotting, added call to GPRARCS.
ATHPRI - Comma added to FORMAT 460 statement.
CATHENA - Main, renamed from ATHENA, open luns 2 and 5, define CPUTYP
CIRC - Plotting, added call to GPRCIRC.
CUROT - Passing double precision constants to CUROOT, 2 places.
FINDMOD - FORMAT(F) changed to runtime format.
FINDPOS - FORMAT(F) changed to runtime format.
FLUXBUG - DLines modified.
GETDEF - FORMAT(I) changed to runtime format.
GPRARCS - Dummy APOLLO routine for plotting arcs.
GPRCIRC - Tektronix or APOLLO plotting, circles, not used.
GPRPIEC - Tektronix or APOLLO plotting, pie chart, not used.
GPRPLOT - Tektronix or APOLLO plotting, including SKPNXT logic.
GPRPLOTS - Tektronix or APOLLO plot initialization.
GPRSYMBOL - Tektronix or APOLLO plotting, including SKPNXT logic.
HEATFLX - IBUGHEA added, .FALSE., REAL*8 constants passed to PRESTR, 2 locs
HTBCURV - EXTERNAL CONVECT statement added.
HTCHFCK - DLines modified.
HTDBPRI - DLines modified, FORMAT statement comma added 2 places.
HTDEBUG - DLines modified.
HTWALLF - Logical variable IBUGWAL added, and declared .FALSE.

TABLE 1 (Continued)
Summary of CATHENA file modifications

IHSLD	- Comment statement placed after subroutine name.
INPCNTL	- FORMATs modified using "DUM" variables to eliminate output *******.
INPIC	- CPUTYP reference for OPEN restart file with large RECL 2 places.
INPMOD1	- Hollerith comparison of 4HTIME modified.
INPTDVA	- CTAGI was not declared character, error on APOLLO.
INPUT	- initialize TITL1C and TITL2C to blanks.
INTBICA	- CALL INTEG argument (1.0-RF) changed to (1.0D0-RF), 3 places.
INTBICB	- CALL INTEG argument (1.0-RF) changed to (1.0D0-RF), 6 places.
INTBICD	- Character concatenation outside argument call list, 2 places.
INWL2HA	- initializing and addressing of NUMCHA(8) invalid in loop 1003 I>8.
INWL4B	- initializing and addressing of NUMCHA(8) invalid in loop 1005 I>8.
INWLBUG	- DLines, PARAMETER comparison of IROUIN and IROUEXE modified.
INWLWAT	- OPEN statement "SHARED" removed 2 places.
PIEC	- Plotting, added call to GPRPIEC, pie charts.
PLOT	- Plotting, added call to GPRPLOT.
PLOTIN	- Added 'CLOSE' INPUT FILE for screen plotting from keyboard.
PLOTS	- Plotting, added call to GPRPLOTS.
PRNTNC	- FORMAT modified, 0PI6 changed to 0P,I6.
PUTSP	- SQRT statement modified, -1.0D0 replaced by variable ZNEG.
SKPNXT	- Skip next logic for GPRPLOT and GPRSYMBOL skip logic.
SPLITH	- DLine added on statement immediately preceeding "1100 FORMAT".
SUMEQ	- Pass REAL*8 constants to HTLOCAL 4 places, and HTWALL one place.
SURGER1	- Comma removed following "16" from statement CHARACTER*16,CHARIN.
SYMBOL	- Plotting, added call to GPRSYMBOL.
TDVMOD2	- Hollerith variables HCONTROL and HRCMODEL added.
TPLOT	- REAL*4 XOFF statement added.
TRIM	- Checks 80 characters for trailing blanks, needs LEN function.
WALL2	- Comma added to FORMAT statement.
WALL6A	- Comma added to FORMAT statement.
WALLPTS	- DLines modified.
WALLPTT	- DLines modified.
WALLPTU	- DLines modified.
WATCH	- OPEN statement "SHARED" removed.

APPENDIX A

CVA (CONVERT VAX TO APOLLO) SOURCE CODE LISTING.

```
      PROGRAM CVA
C     THIS ROUTINE CONVERTS VAX FORTRAN SOURCE FILES TO APOLLO FTN FILES.
C     IN THE FOLLOWING SEQUENCE:

C     Ø) ALL FILES TO BE SENT ARE CONCATENATED INTO ONE CONTINUOUS FILE
C           CAPTURED BY THE APOLLO PROGRAM 'EMT'. EACH FILE PASSED IS
C           GIVEN THE HEADER HEADERFILENAME 'FILENAME.EXTENSION'.
C           THE FIRST FILE PASSED IS A FORTRAN PROGRAM FOR READING
C           THE LARGE FILE AND EXTRACTING THE SOURCE LINES.

C     1) COMMENT CARDS ARE BYPASSED

*     2) TABS APPEARING BEFORE COLUMN 6 ARE CONVERTED TO SPACES, SO THAT
*           THE NEXT CHARACTER APPEARS IN COLUMN 7. IF THE CHARACTER
*           FOLLOWING THE TAB IS A DIGIT BETWEEN 1 AND 9, A CONTINUATION
*           LINES HAS OCCURRED, AND THE PRECEDING SPACE IS DELETED SO THAT
*           THE DIGIT WILL OCCUPY COLUMN 6. THE RECORD LENGTH IS THEN
*           CHECKED SO AS NOT TO EXCEED COLUMN 72. THE RECORD LENGTH IS
*           THEN CHECKED SO AS NOT TO EXCEED COLUMN 72. IF IT DOES, THE
*           EXCESS IS WRITTEN ON THE NEXT CONTINUATION LINE.

C     3) INCLUDE STATEMENTS ARE CONVERTED TO "%INCLUDE"  (NO OP ON DINCLUDE)

C     4) ALL LOGICAL DIRECTORIES IN INCLUDE STMTS ARE REMOVED. (THE
C           TEXT BETWEEN THE FIRST ' AND THE : TERMINATOR.

C     5) D LINES ARE MAINTAINED (EXCEPT FOR DINCLUDE).

C     6) NULL LINES ARE CONVERTED TO A 'C' COMMENT LINE

C     7) ALL "ACCEPT *," STATEMENTS ARE CHANGED TO "READ(5,*)"

C     8) ALL "!" IN-LINE COMMENTS CHANGED TO "{"

C     9) MULTIPLE SUBROUTINES, FUNCTIONS BLOCK DATA, OR PROGRAM IN
C           A DECK ARE GIVEN A HEADER
```

CVA (CONVERT VAX TO APOLLO) SOURCE CODE LISTING.

```
      INTEGER OUTFILE
      CHARACTER*30 INNAME,OUTNAME,WORD*10,SIGNAL,OUTNAMEL
      CHARACTER*200 INPC
      BYTE BTAB
      LOGICAL*1 APOLLO,COMMENT,OPN,INLINE,DLINE,FAST
      CHARACTER*1 INP(200),BLANK,TAB,CONTINU(6),NULL,PRIME
      EQUIVALENCE (TAB,BTAB),(NULL,BNULL)
      DATA CONTINU/5*' ','_'/
      DATA BLANK/' '/
      DATA BTAB/'09'X/,BNULL/'00'X/,PRIME/'27'X/

C     SET UP TWO FILES FOR THE APOLLO, THE FTN AND BIND COMMAND FILES
C           WHICH WILL CONTAIN ALL FILE NAMES TRANSFERRED.

      IFTN=13
      IBIND=14
      OPEN(UNIT=IFTN,FILE='F.COM',STATUS='NEW'
     1     ,CARRIAGECONTROL='LIST')
      OPEN(UNIT=IBIND,FILE='B.COM',STATUS='NEW'
     1     ,CARRIAGECONTROL='LIST')

C     SET UP ALL OTHER FILES
      INFILE=11
      OUTFILE=12
      INDECK=5                  !INPUT UNIT FOR FILES TO BE TRANSFERED
      IPRNT=6


      NOPER=0
      NFILES=0
      IREAD=0
      ICOUT=0
      APOLLO=.TRUE.
      OPN=.FALSE.
      GOTO 44

1     CONTINUE
C     ERROR OPENING A SOURCE FILE
      IF(.NOT.APOLLO) TYPE 77, INNAME
77    FORMAT(' ERROR OPENING FILE ',A,'  BYPASSING')
      NOPER=NOPER+1

2     CONTINUE
C     CLOSE THE COMPLETED FILES
      OPN=.FALSE.
      IF(NFILES.GT.0) THEN
      CLOSE(UNIT=INFILE)
      IF(.NOT.APOLLO) THEN
      CLOSE(UNIT=OUTFILE)
      ELSE
```

CVA (CONVERT VAX TO APOLLO) SOURCE CODE LISTING.

```
C       SEND CTRL Q TO APOLLO TO TERMINATE LAST FILE ON EMT
C       TYPE 888, '11'X
C888    FORMAT(1X,A)
        ENDIF
        ENDIF

44      CONTINUE
C       GET THE VAX INPUT FILE NAME.
        IF(.NOT.APOLLO) TYPE 3
3       FORMAT('  INPUT FILE NAME ?',$)
        READ(INDECK,12,END=1000)  NCH,INNAME
        IF(NCH.EQ.0) GOTO 44
C       STRIP OFF THE FILE VERSION NUMBER ON THE VAX, AND OPEN THE FILE.
        NCH=INDEX(INNAME,';')
        IF(NCH.EQ.0) GOTO 44
        NCH=NCH-1
        OPEN(UNIT=INFILE,FILE=INNAME(1:NCH),STATUS='OLD',ERR=1)

C       DETERMINE WHETHER TO PERFORM A FAST TRANSFER
        FAST=.TRUE.
        IF(INNAME(NCH-2:NCH).EQ.'FOR'
     1        .OR.INNAME(NCH-2:NCH).EQ.'CMN'
     1        .OR.INNAME(NCH-2:NCH).EQ.'PLT'
     1        .OR.INNAME(NCH-2:NCH).EQ.'VAX'
     1        .OR.INNAME(NCH-2:NCH).EQ.'CDC') FAST=.FALSE.

        IF(.NOT.APOLLO) THEN
C       OPEN A NEW OUTPUT FILE IF NOT AN APOLLO TRANSFER.
        OPEN(UNIT=OUTFILE,FILE=INNAME(1:K1),STATUS='NEW'
     1        ,CARRIAGECONTROL='LIST')
        OPN=.TRUE.
        NFILES=NFILES+1
        ENDIF
10      CONTINUE
C       READ A SOURCE CODE LINE ON THE VAX.
        READ(INFILE,12,END=2) MAXCOL,INPC
        IREAD=IREAD+1
12      FORMAT(Q,A)

        IF(FAST) GOTO 18

C       CHECK FOR 'SUBROUTINE', 'PROGRAM', 'BLOCK DATA', OR 'FUNCTION'
C               CARDS AND OPEN NEW FILE IF NOT ALREADY OPENED.
        IF(APOLLO) THEN
        K2=INDEX(INPC(1:MAXCOL),'SUBROUTINE ')
        IF(K2.EQ.0) K2=INDEX(INPC(1:MAXCOL),'FUNCTION ')
        IF(K2.EQ.0) K2=INDEX(INPC(1:MAXCOL),'PROGRAM ')
        IF(K2.EQ.0) K2=INDEX(INPC(1:MAXCOL),'BLOCK DATA ')
        IF(K2.LE.1) GOTO  18
C       LEADING CHARACTERS MUST NOT BE PRESENT, EG, COMMENTS, ELSE BYPASS
        DO I=1,K2-1
        IF(INPC(I:I).NE.' '.AND.INPC(I:I).NE.TAB) GOTO 18
        ENDDO
```

```
C       BYPASS THE FIRST BLANK IN THE BLOCK DATA STMT
        K2=K2+6
C       LOCATED NEW PROGRAM, SUBROUTINE OR FUNCTION, LOCATE THE BLANK
C               TO POSITION BEFORE THE NAME OF THE ROUTINE
        K3=INDEX(INPC(K2:),' ')
        K2=K2+K3
C       ADVANCE THE POINTER PAST BLANKS TO THE SUBROUTINE NAME
        DO I=K2,MAXCOL
        IF(INPC(I:I).NE.' '.AND.INPC(I:I).NE.TAB) GOTO 16
        ENDDO
        GOTO 18          !BYPASS IF NO NAME PRESENT
16      CONTINUE
        K2=I
C       LOCATE THE OPENING '(' FOR ARGUMENT LIST, OR THE FIRST TRAILING BLANK
C       OR TAB AS TERMINATOR OF THE NAME
        K4=INDEX(INPC(K2:),'(')
        K5=INDEX(INPC(K2:),' ')
        K6=INDEX(INPC(K2:),TAB)
C       CHOOSE THE FIRST TERMINATOR OF THE NAME
        IF(K6.GT.0.AND.K6.LT.K5) K5=K6
        IF(K5.GT.0.AND.K5.LT.K4) K4=K5
        IF(K4.EQ.0) K4=K5
        IF(K4.EQ.0) K4=MAXCOL-K2+2
C       SET THE NAME LOCATION AND NUMBER OF CHARACTERS
        K4=K2+K4-2
        NC=K4-K2+1
        OUTNAME=INPC(K2:K4)
C       CHECK THAT THE FILE HAS NOT ALREADY BEEN OPENED, EG BY
C       LEADING COMMENT STATEMENTS BEFORE THE ROUTINE NAME.
        IF(OUTNAMEL(1:NC+4).EQ.OUTNAME(1:NC)//'.FTN') GOTO 18
        OUTNAMEL=OUTNAME(1:NC)//'.FTN'
C       SEND THE NEW FILE HEADER TO THE APOLLO
        TYPE 203, OUTNAME(1:NC)//'.FTN'
        OPN=.TRUE.
C       RECORD THE FTN AND BIND FILE NAMES
        WRITE(IFTN,204) OUTNAME(1:NC)
        WRITE(IBIND,205) OUTNAME(1:NC)
        ENDIF
18      CONTINUE
C       CHECK FOR 'CMN' DECKS ETC, AND OPEN ORIGINAL FILE NAME IF NOT
C       OPENED BY SUBROUTINE, PROGRAM OR FUNCTION CALL.
C       THE FORTRAN FILES FROM TWO VAX SUBDIRECTORIES ([.CDC] AND [.PLOT]) WERE
C       NAMED TO *.CDC AND *.PLT. THESE FILES ARE RENAMED TO .FTN IN THE
C       TRANSFER TO THE APOLLO IN THE NEXT 'IF' STATEMENTS.
C
        IF(.NOT.OPN) THEN
        OUTNAME=INNAME
        IF(INNAME(NCH-2:NCH).EQ.'FOR') OUTNAME(NCH-2:NCH)='FTN'
        IF(INNAME(NCH-2:NCH).EQ.'PLT') OUTNAME(NCH-2:NCH)='FTN'
        IF(INNAME(NCH-2:NCH).EQ.'VAX') OUTNAME(NCH-2:NCH)='FTN'
        IF(INNAME(NCH-2:NCH).EQ.'CDC') OUTNAME(NCH-2:NCH)='FTN'
        TYPE 203, OUTNAME(1:NCH)
```

```
        OPN=.TRUE.
        OUTNAMEL=OUTNAME
        IF(OUTNAME(NCH-2:NCH).NE.'FTN') GOTO 28
        WRITE(IFTN,204) OUTNAME(1:NCH-4)
        WRITE(IBIND,205) OUTNAME(1:NCH-4)
        ENDIF

28      CONTINUE

        IF(FAST) THEN
        IF(MAXCOL.GT.132) MAXCOL=132
        IF(APOLLO) THEN
        TYPE 21, INPC(1:MAXCOL)
21      FORMAT(1X,A)
        ELSE
        WRITE(OUTFILE,21) INPC(1:MAXCOL)
        ENDIF
        ICOUT=ICOUT+1
        GOTO 10
        ENDIF
        NC=0
        I1=0

C       FILE TYPES MAY BE FORTRAN OR TEXT, EG '.COM' FILES. IF FILES
C       ARE '.COM' ($ IN COL1), TREAT EACH RECORD AS A COMMENT

C       SEARCH FOR COMMENT CARD AND FLAG AS COMMENT
C       NC AND I1 ARE POINTERS FOR LOCATION IN ARRAYS INP AND INPC RESP.
C       NC POINTS TO THE LAST CHARACTER INSERTED INTO INP,
C       I1 POINTS TO THE NEXT CHARACTER TO BE CONSIDERED IN INPC.
        COMMENT=.FALSE.
        IF(INPC(1:1).EQ.'C'.OR.INPC(1:1).EQ.'c'
        1  .OR.INPC(1:1).EQ.'*'.OR.INPC(1:1).EQ.'$') THEN
        NC=1
        I1=1
        INP(NC)=INPC(1:1)
        COMMENT=.TRUE.
        IF(INP(1).EQ.'*') INP(1)='C'
        I1=I1+1
        GOTO 446
        ENDIF

C       SEARCH FOR NULL LINES AND INSERT A 'C' IN COLUMN 1.
        IF(MAXCOL.EQ.0) THEN
        INP(1)='C'
        NC=1
        GOTO 777
        ENDIF

C       SEARCH FOR  VAX "!" IN-LINE COMMENTS AND CHANGE TO "{"
        INLINE=.FALSE.
        KK=INDEX(INPC,'!')
```

```
            IF(KK.GT.Ø) THEN
            INPC(KK:KK)='{'
            INLINE=.TRUE.
            ENDIF
C           SEARCH FOR "INCLUDE '" AND CONVERT TO %INCLUDE ' STARTING IN COLUMN 1,
C           UNLESS THIS IS A 'DLINES' CARD WHERE IT WILL GO INTO COLUMN 2.
C           THEN SEARCH FOR ":" AND REMOVE THE VAX LOGICAL VARIABLE REFERENCED
C           BETWEEN THE ' AND THE : EG, "INCLUDE 'SSS:DECK.CMN'"

            K1=INDEX(INPC(2:15),'INCLUDE ')
            IF(K1.LE.Ø) GOTO 444
            IF(INPC(1:1).EQ.'D'.OR.INPC(1:1).EQ.'d') THEN
            INP(1)=INPC(1:1)
            NC=1
C           APOLLO DOES NOT SUPPORT LEADING "D" ON %INCLUDE, THEREFORE, SET NC TO Ø.
            NC=Ø
            ENDIF
            INP(NC+1)='%'
            INP(NC+2)='I'
            INP(NC+3)='N'
            INP(NC+4)='C'
            INP(NC+5)='L'
            INP(NC+6)='U'
            INP(NC+7)='D'
            INP(NC+8)='E'
            INP(NC+9)=' '
            INP(NC+1Ø)=PRIME
            NC=NC+1Ø

C           SEARCH FOR SINGLE QUOTE, THEN : AND COPY THE CHARACTERS AFTER
            K1=INDEX(INPC(8:MAXCOL),PRIME)
            K2=INDEX(INPC(8:MAXCOL),':')

C           MOVE THE K2 POINTER TO THE CHARACTER FOLLOWING THE ' OR :
            IF(K2.EQ.Ø) THEN
            I1=K1+7+1
            ELSE
            I1=7+K2+2
            ENDIF

C           CHECK FOR THE VAX "/[NO]LIST" PARAMETER, AND DELETE BY INSERTING
C                  THE FINAL ' AT THE "/", AND REDUCING THE CHARACTER COUNT
C                  IN THE ORIGINAL STRING INPC.
            K3=INDEX(INPC(8:MAXCOL),'/LIST')
            IF(K3.EQ.Ø) K3=INDEX(INPC(8:MAXCOL),'/NOLIST')
            IF(K3.GT.Ø) THEN
            MAXCOL=K3+7
            INPC(MAXCOL:MAXCOL)=PRIME
            ENDIF
C           NOW TRANSFER CHARACTERS REMAINING, AFTER THE : TO THE '
            GOTO 446
```

```
444     CONTINUE
C       CHECK FOR A "D" CARD
        I1=0
        NC=0
        DLINE=.FALSE.
        IF(INPC(1:1).EQ.'D'.OR.INPC(1:1).EQ.'d') THEN
        NC=1
        INP(NC)=INPC(1:1)
        I1=1
        DLINE=.TRUE.
        ENDIF

C       REPLACE ALL LEADING TABS WITH BLANKS TO COLUMN 6.
        I1=I1+1
        IF(INPC(I1:I1).NE.TAB) GOTO 446
        DO I=I1,6
        NC=NC+1
        INP(NC)=BLANK
        ENDDO
        I1=I1+1

C       CHECK FOR A TABBED CONTINUATION CARD, AND MOVE POINTER TO COLUMN 5
        IF(INPC(I1:I1).GE.'1'.AND.INPC(I1:I1).LE.'9'.AND.I1.LE.MAXCOL)
        1       NC=NC-1

446     CONTINUE
C       SEARCH FOR "ACCEPT *," STATEMENTS AND CHANGE TO "READ(5,*)"
        KK=INDEX(INPC,'ACCEPT *,')
        IF(KK.GT.0) INPC(KK:KK+8)='READ(5,*)'

C       FOLLOWING LOGIC TO STATEMENT LABEL 604 WAS NOT REQUIRED SINCE
C               THE ASSOCIATED ROUTINES WERE CONVERTED TO RUN TIME FORMATS.
CC      SEARCH FOR "FORMAT(F)" STATEMENTS AND CHANGE TO "FORMAT(F15.3)"
C       KK=INDEX(INPC,'FORMAT(')
C       IF(KK.EQ.0) GOTO 604
C       KK=INDEX(INPC,'FORMAT(F)')
C       IF(KK.GT.0) THEN
CC      TYPE 303, OUTNAME(1:NC),INPC(1:MAXCOL)
CC303   FORMAT(' WARNING, FILE',A,' STATEMENT',A,/
CC      1 ' IS BEING CONVERTED TO ',$)
C       INPC(KK:KK+12)='FORMAT(F15.3)'
C       MAXCOL=MAXCOL+4
CC      TYPE 304, INPC(1:MAXCOL)
CC304   FORMAT(1X,A)
C       GOTO 604
C       ENDIF
C
C       KK=INDEX(INPC,'FORMAT(I)')
C       IF(KK.GT.0) THEN
CC      TYPE 303, OUTNAME(1:NC),INPC(1:MAXCOL)
C       INPC(KK:KK+10)='FORMAT(I10)'
C       MAXCOL=MAXCOL+2
```

```
CC      TYPE 304, INPC(1:MAXCOL)
C       GOTO 604
C       ENDIF
C
C604    CONTINUE


C       SEARCH REMAINING INPUT STMT FOR TABS AND CONVERT EACH TO 6 SPACES
        DO I=I1,MAXCOL
        IF(INPC(I:I).EQ.TAB) THEN
        JFILL=6
        IF(NC.LT.6) JFILL=6-NC
        DO J=1,JFILL
        NC=NC+1
        INP(NC)=BLANK
        ENDDO
        ELSE
C       COPY EACH CHARACTER INTO THE NEW OUTPUT RECORD
        NC=NC+1
        INP(NC)=INPC(I:I)
        ENDIF
        ENDDO



777     CONTINUE
C       OUTPUT THE FIRST 72 CHARACTERS ON THE STMT, UNLESS THIS IS A COMMENT
C       WHERE WE WILL OUTPUT ONLY 79 CHARACTERS TO PREVENT A NULL LINE
C       ON THE APOLLO FOLLOWING AN 80 CHARACTER STMT.
C       TRAILING CHARACTERS ON COMMENT STATEMENTS AND INLINE COMMENTS WILL
C       BE LOST.
        J=NC
        IF(J.GT.79) J=79
        IF(J.GT.72.AND..NOT.COMMENT.AND..NOT.INLINE) J=72
778     CONTINUE
        IF(APOLLO) THEN
        TYPE 11, (INP(I),I=1,J)
        ELSE
        WRITE(OUTFILE,11) (INP(I),I=1,J)
        ENDIF
        ICOUT=ICOUT+1

C       OUTPUT THE REMAINING CHARACTERS OF THIS STMT
        IF(COMMENT.OR.INLINE) GOTO 10   !BYPASS LAST CHARACTERS ON COMMENT
        IF(NC.GT.J) THEN
C       IF THIS IS A DLINE, CONTINUE NEXT LINE WITH A LEADING 'D'.
        CONTINU(1)=' '
        IF(DLINE) CONTINU(1)='D'
        IF(APOLLO) THEN
        TYPE 11, CONTINU,(INP(I),I=J+1,NC)
        ELSE
        WRITE(OUTFILE,11) CONTINU,(INP(I),I=J+1,NC)
```

```
        ENDIF
        ICOUT=ICOUT+1
        ENDIF
        GO TO 10

1000    CONTINUE

C       SEND A NULL LINE DOWN TO CLOSE OUT LAST GOOD LINE AT APOLLO
        TYPE 11, ' '

C       CLOSE THE COMMAND PROCEDURES
        WRITE(IBIND,208)
        CLOSE(UNIT=IFTN)
        CLOSE(UNIT=IBIND)

        IF(.NOT.APOLLO) TYPE 999,NFILES,IREAD,ICOUT
999     FORMAT(' CONVERTED ',I5,' FILES, READ 'I10,' LINES',
        1', WROTE ',I10,' LINES')

        IF(NOPER.GT.0) THEN
        IF(.NOT.APOLLO) TYPE 1001, NOPER
1001    FORMAT(' ERROR OPENING ',I5,
        1' FILES, INTENTIONAL INTEGER DIVIDE BY 0')
        NOPER=0
        I=1/NOPER
        ENDIF


11      FORMAT(1X,200A1)
203     FORMAT(1X,'HEADERFILENAME ',A)

C       APOLLO 'FTN' AND 'BIND' FILE DEFINITIONS-- EACH FILE NAME
C              IS WRITTEN TO EACH OF THESE FILES.
C       COND = DLINES COMPILE SWITCH, NCOND = NO DLINE COMPILE
C       NPEB = NO PERFORMANCE ENHANCMENT BOARD, or -CPU 3000
C       SAVE = STATIC ROUTINE VARIABLES
C       ZERO = ZERO MEMORY INITIALLY
C       OPT = OPTIMIZE CODE
C       INDEXL = OPTIMIZE SUBSCRIPT COMPUTATION
C204    FORMAT('FTN 'A,'.FTN -COND -SAVE -ZERO -INDEXL -OPT -CPU 3000')
204     FORMAT('FTN 'A,'.FTN -COND -SAVE -NPEB -ZERO -INDEXL')
205     FORMAT(A,'.BIN')
208     FORMAT('-B CATHENA'/' -END')
        END
```

APPENDIX B

TRANSFER OF FILES FROM VAX TO APOLLO

B.Ø.1 File Transfer Modes

The transfer of files from the VAX to APOLLO was achieved using either magnetic tape or the APOLLO EMT (EMulate Terminal) utility. A total of approximately 5ØØ source files (.FTN and .CMN) having approximately 5Ø,ØØØ lines of coding were transferred and captured into file C.FOR. Each file was labelled with a special header. Wardrop's SPLIT program was used on the APOLLO to extract each individual file from C.FOR into its designated name. The C.FOR file was edited to remove initial and final garbage transferred from the VAX.

B.Ø.2 Magnetic Tape Using RWMT

A 16ØØ bpi 9 track magnetic tape was created on the VAX using the COPY command which generates ANSI 4 format. The files placed on the tape were the list of files MT.LIS, the Fortran source files (including SPLIT) concatenated into a single file C.FOR, the 3 command procedure files for compiling and binding the source code, and a series of 19 files for verification (input and VAX output). This tape can be read on the APOLLO using RWMT. A sample APOLLO command for reading the magnetic tape follows:

```
RWMT -DEV MØ -BL 12ØØØ -ANSI -F 1 -R     /CATHENA/MT.LIS
```

The file MT.LIS contains a list of the remaining files on the magnetic tape. A second command reads the remaining files starting at file location 2 using the file names listed in MT.LIS.

```
RWMT -DEV MØ -BL 12ØØØ -ANSI -F 2 -R     */CATHENA/MT.LIS
```

TRANSFER OF FILES FROM VAX TO APOLLO

B.Ø.3 Cartridge Tape Using RBAK

The 1600 bpi magnetic tape was read on an APOLLO computer and transfered to cartridge tape for the target APOLLO 3000 Work Station. RWMT was used to transfer files from the magnetic tape to disk, and WBAK was used to transfer the disk files to cartridge tape. The RWMT command used was defined above. The WBAK command used was:

```
WBAK -DEV CT -F 1 -L -RETEN -FULL /CATHENA?*
```

The command to retrive the files from cartridge tape is

```
RBAK -DEV CT -F 1 -L -ALL -AS /CATHENA
```

B.Ø.4 Communications With VAX Using EMT

The parameters used for the APOLLO EMT utility for communicating with the VAX were:

```
$EMT
EMT>LINE 1
EMT> TCTL -SPEED 19200 -NLD 20000 -NOECHO
EMT> BREAK 20000
EMT> INTERM VAX
EMT> RCV C.FOR      {Contents will be stored in file C.FOR.
<F1>                {Set APOLLO EMT to remote
LOGIN               {login sequence on VAX, and setting NOECHO.
@SEND               {VAX command procedure to initiate and transfer
LOGOUT              {logout on VAX
<F1>                {Set APOLLO EMT to local
CLOSE               {CLose C.FOR on the APOLLO when done.
```

The file C.FOR will have to be edited to remove garbage at the start and end of the file associated with login and logout.

The VAX command procedure used to prepare the source files for transfer to the APOLLO is listed as follows:

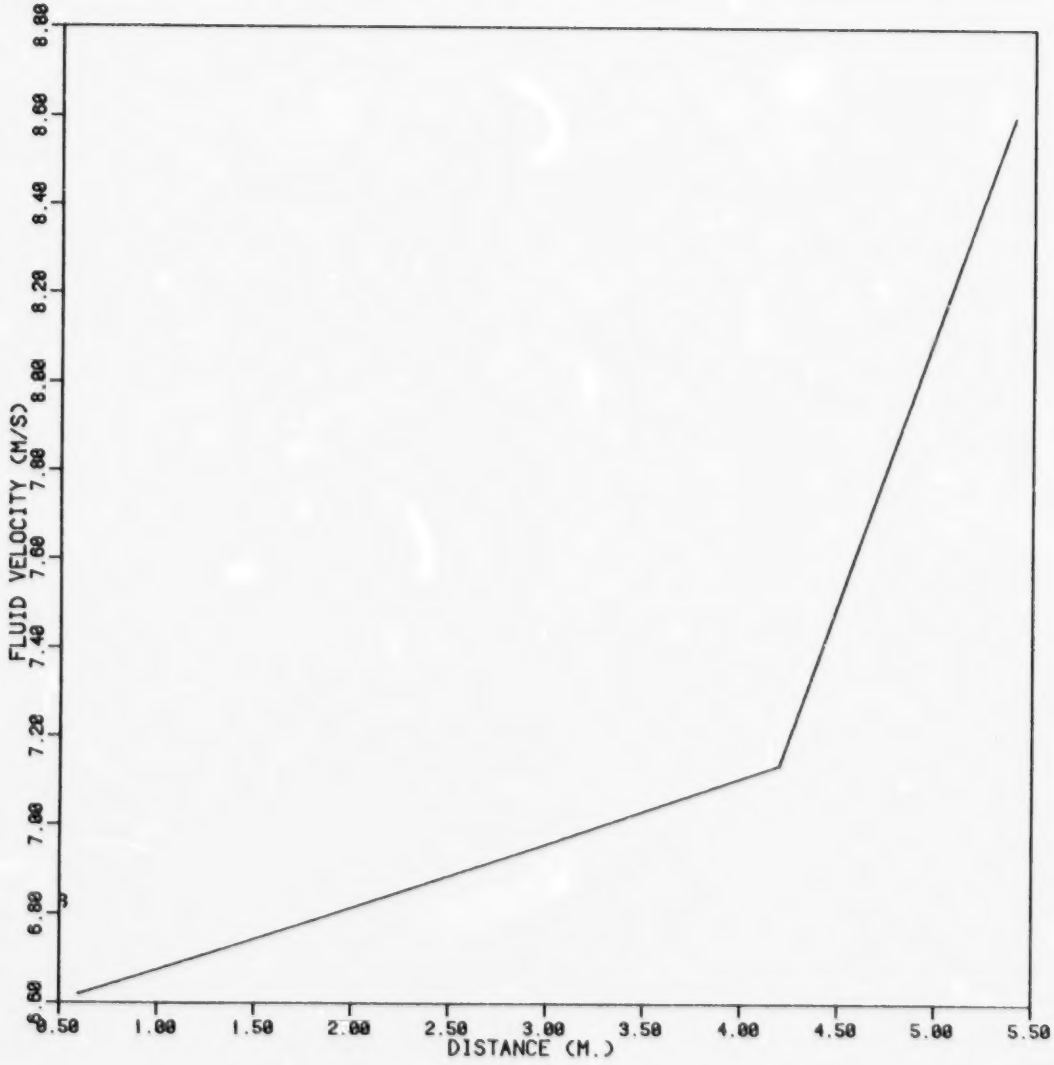
```
$SET NOVER
$!SET TERM/NOECHO
$!DEASS FOR005
$!DEASS FOR006
```

TRANSFER OF FILES FROM VAX TO APOLLO

```
$!
$! TRANSFER THE TWO MAIN WORKING PROGRAMS
$!  (COMBINING FILES ON VAX, AND
$!    SPLITTING ON THE APOLLO).
$!
$SET DEF DUAØ:[CATHENA]
$!COPY MT.LIS MSAØ:*/L
$DIR/SIZ/OUT=F.LIS SPLIT.FOR;,CVA.FOR;
$ASS F.LIS FORØØ5
$ASS F.OUT FORØØ6
$RUN [CATHENA]CVA
$DEASS FORØØ5
$DEASS FORØØ6
$DEL F.COM;*,B.COM;*,F.LIS;*
$!
$SET DEF [CATHENA.SOURCE]
$REN [.CDC]*.FOR; []*.CDC
$REN [.PLOT]*.FOR; []*.PLT
$!
$DIR/SIZ/OUT=F.LIS *.FOR;,*.CMN;,*.CDC;,*.PLT;
$!
$ASS F.LIS FORØØ5
$ASS F.OUT FORØØ6
$RUN [CATHENA]CVA
$DEASS FORØØ5
$DEASS FORØØ6
$!
$DEL F.LIS;*
$REN *.PLT [.PLOT]*.FOR
$REN *.CDC [.CDC]*.FOR
$REN F.COM; FFF.COM
$REN B.COM; BBB.COM
$!
$!DIR/SIZ/OUT=F.LIS FTN.COM;,BIND.COM;
$!ASS f.LIS FORØØ5
$!RUN [CATHENA]CVA
$!DEL F.COM;
$!DEL B.COM;
$!
$!SET DEF [CATHENA.SOURCE.EXAMPLES]
$!DIR/SIZ/OUT=F.LIS EX%.*
$!RUN [CATHENA]CVA
$!DEL F.COM;
$!DEL B.COM;
```

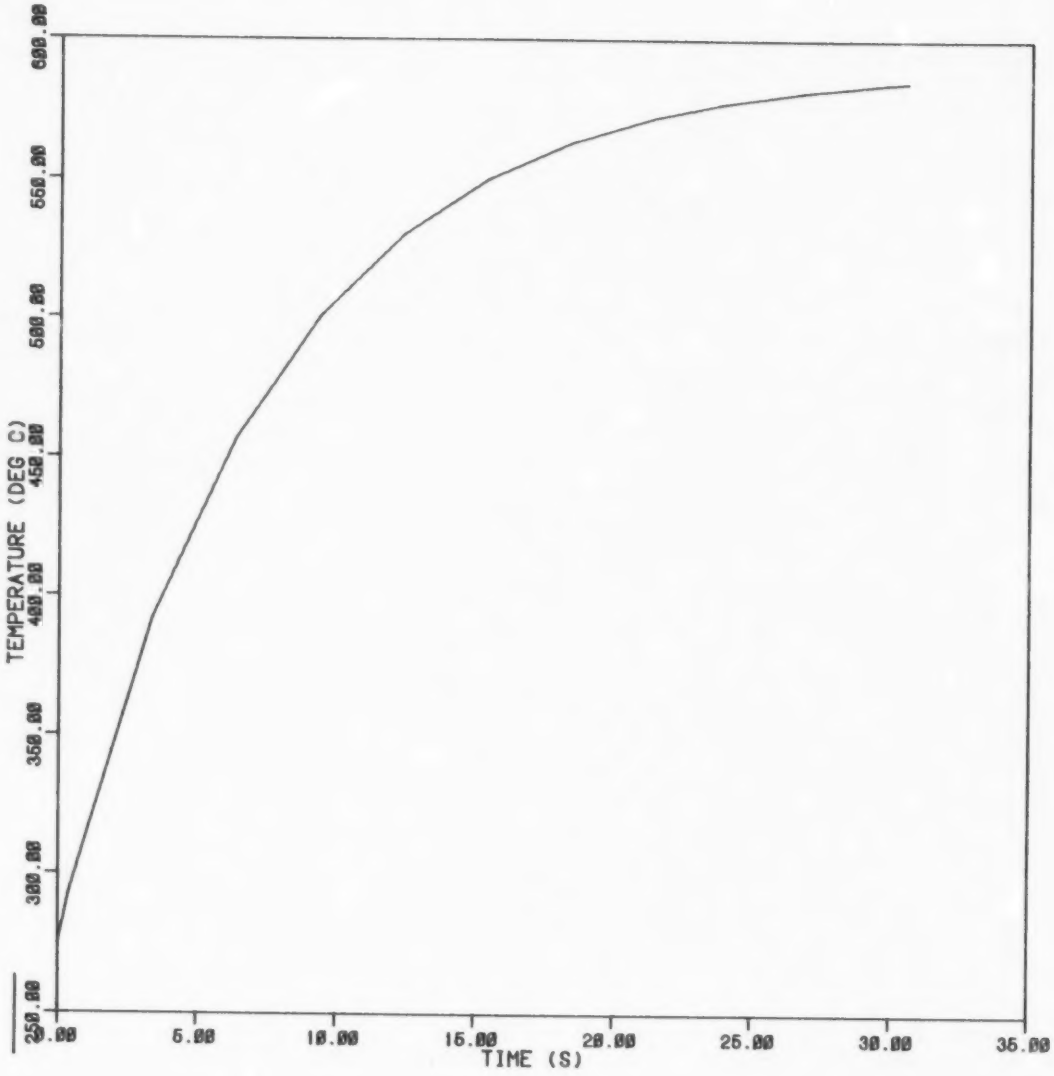
APPENDIX C

SAMPLE TEKTRONIX INTERACTIVE PLOTS

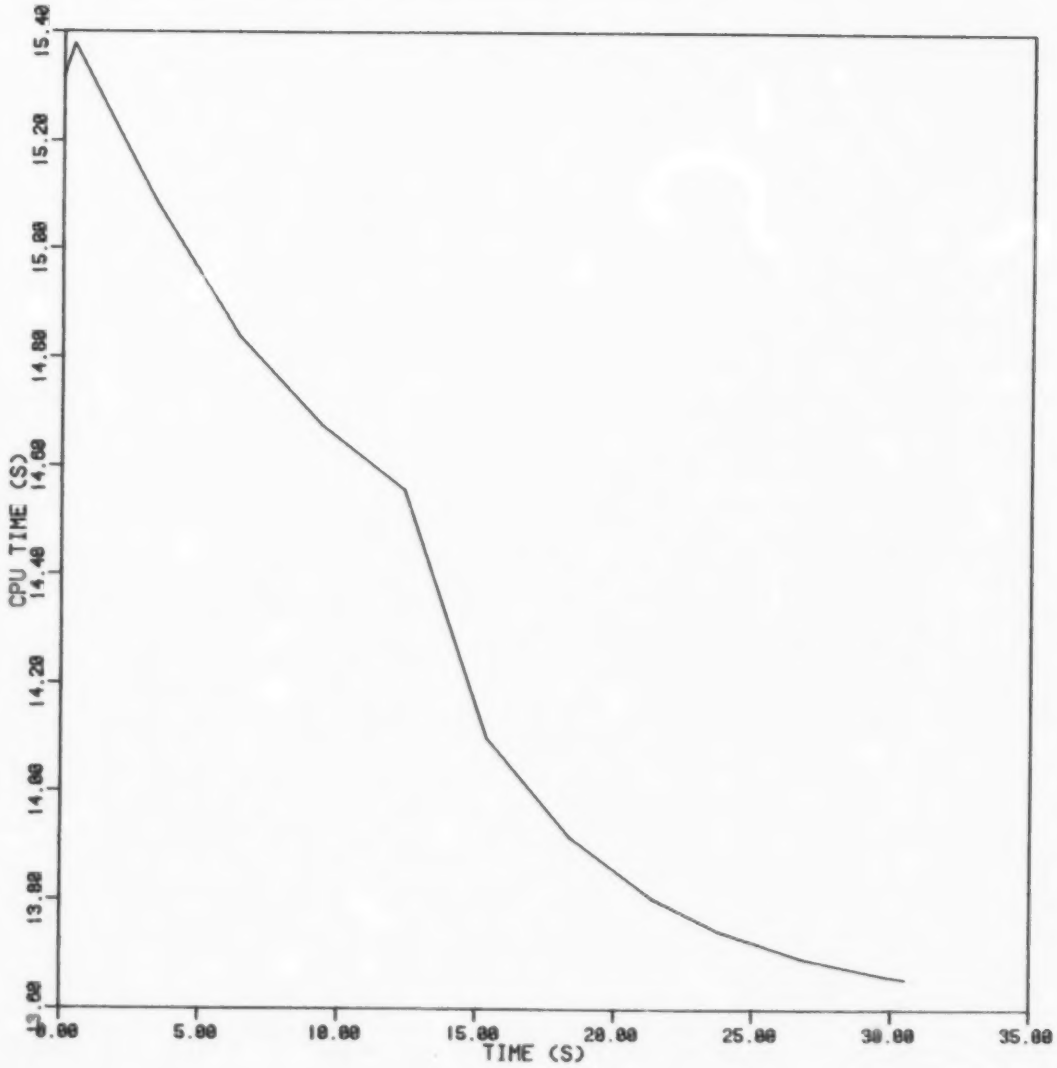


37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:26 ATHENA-MOD 1.2A ON HOST AP3000
AXIAL PLOT FOR BRANCH HTCH @ TIME: 30.530 SEC

IN-OUT PIN 1,IN PIN 10

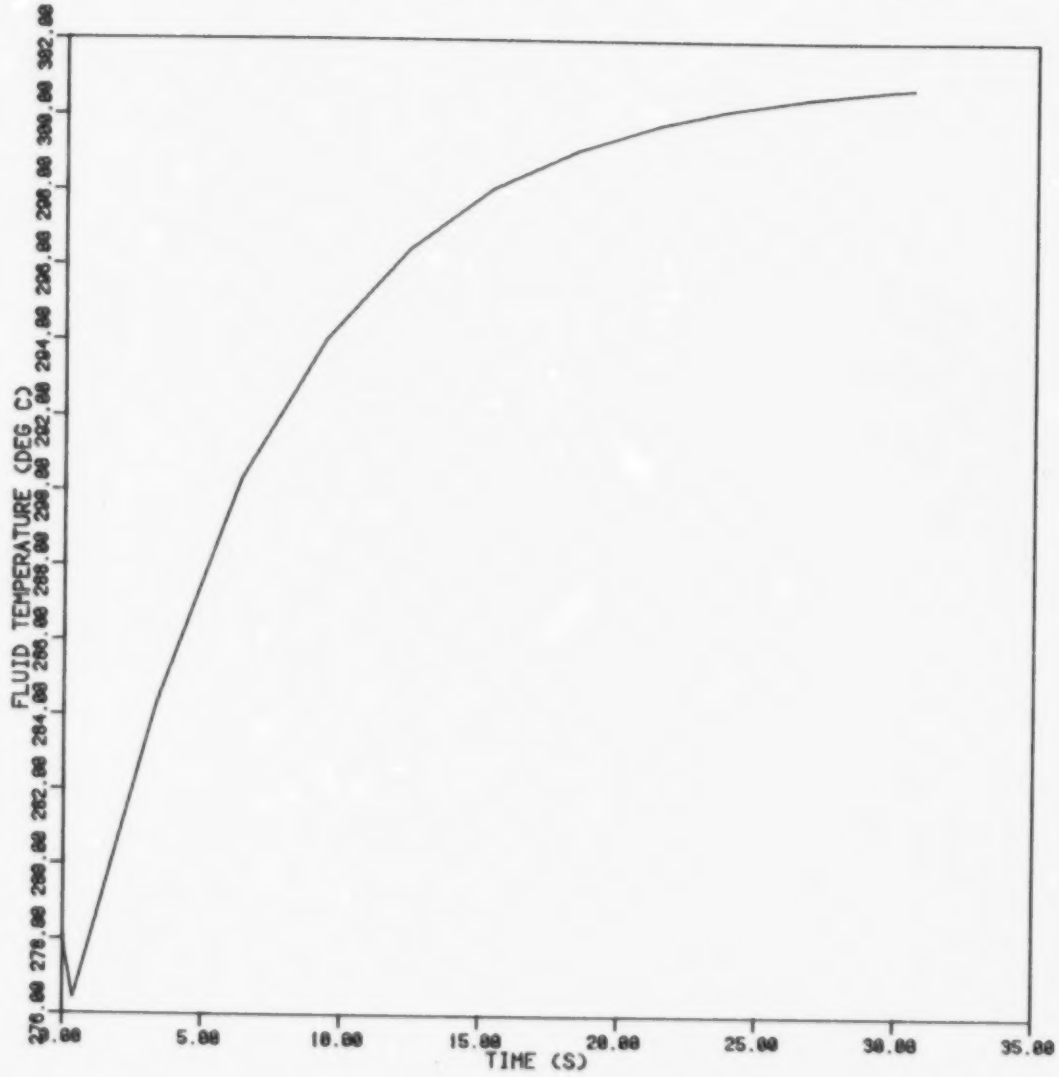


37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:38 ATHENA-MOD 1.2A ON HOST AP3000
MODEL PINS VOLAVG FOR SEGMENT 4 SECTOR 1 PIN GROUP 1 RADIAL NODES 1- 9

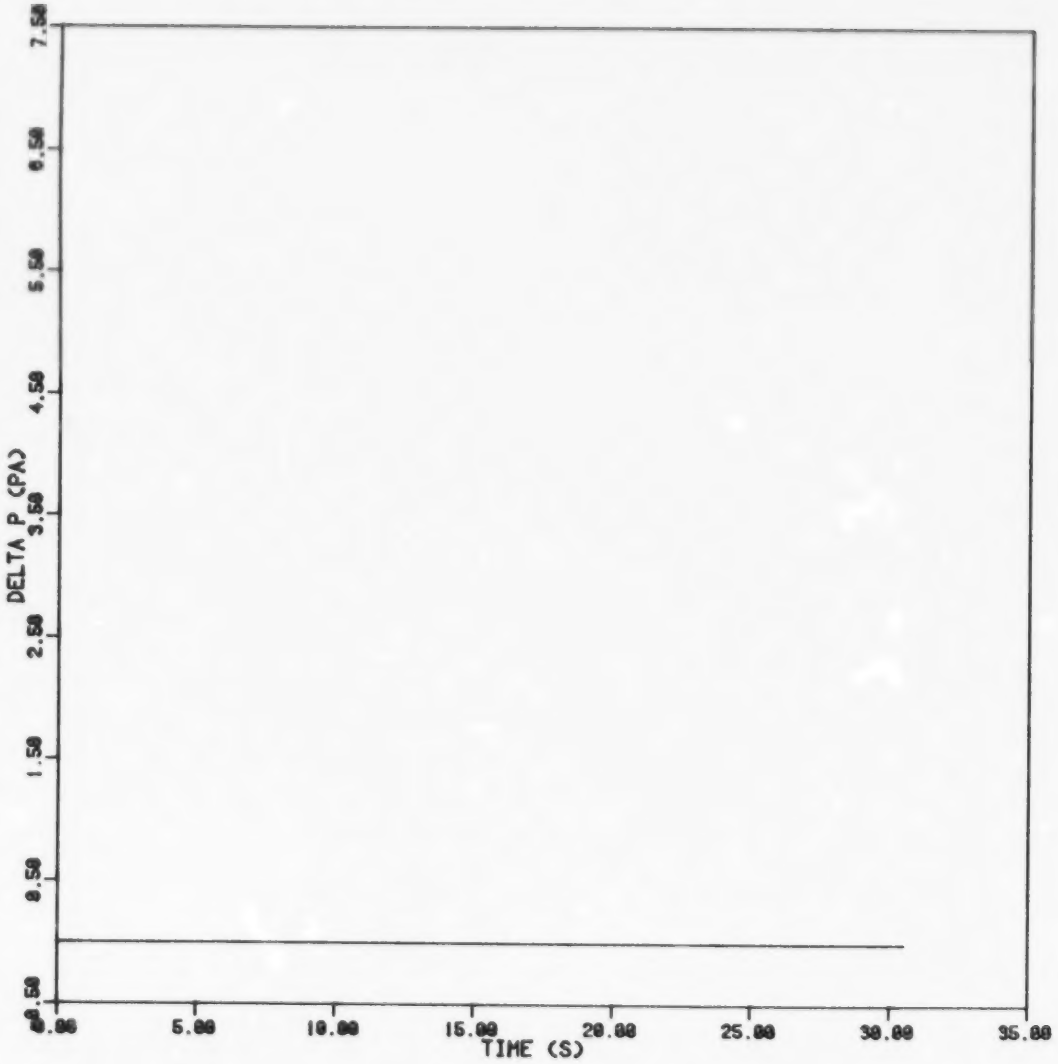


37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:20 ATHENA-MOD 1.2A ON HOST AP3000
CP USAGE VS. SIMULATION TIME

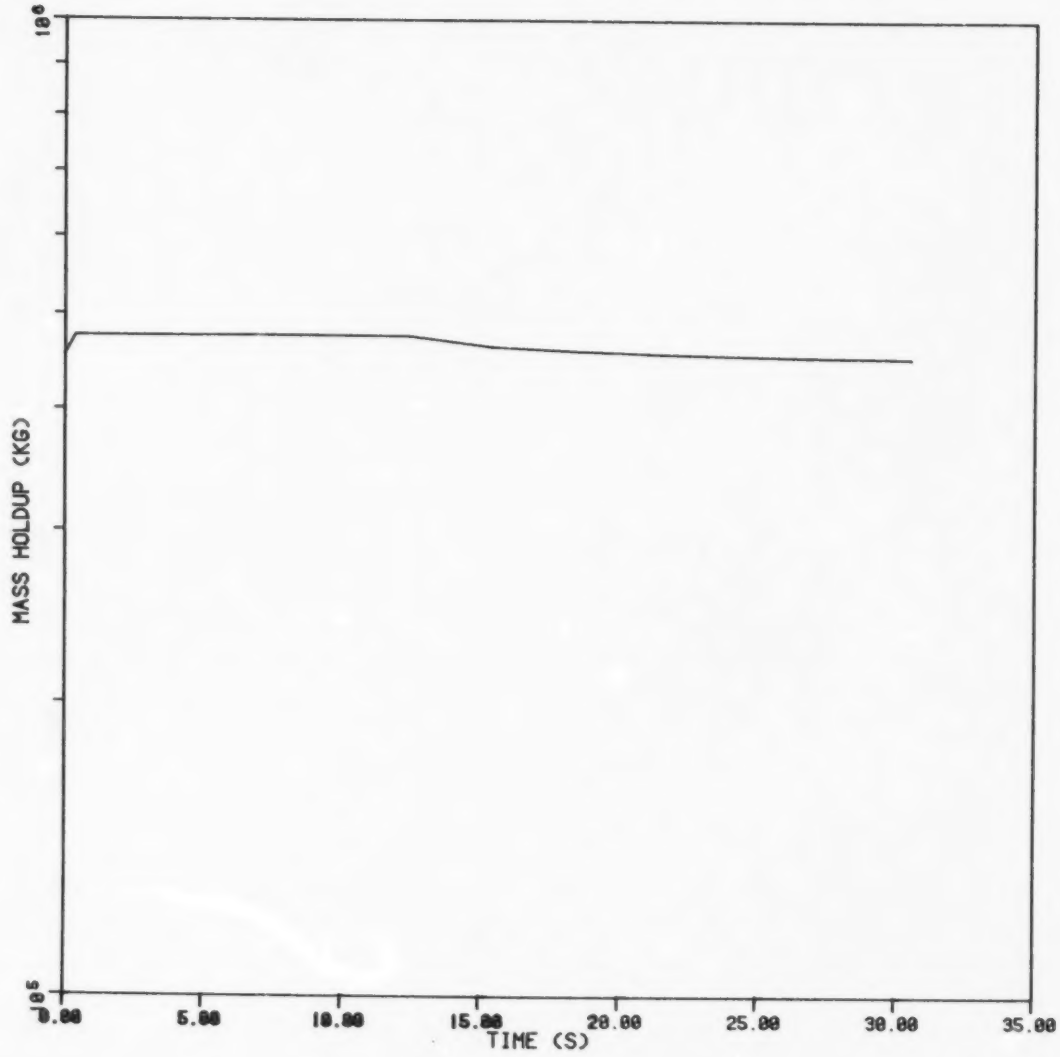
FLUID TEMP MID CHANNEL



37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:26 ATHENA-MOD 1.2A ON HOST AP3000
3.000 M. INTO BRANCH HTCH



37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:26 ATHENA-MOD 1.2A ON HOST APS888
DELTA P FROM 0.100 TO 5.900 M. IN BRANCH HTCH



37-ELEMENT PIN BUNDLE TO TEST CODING IMPLEMENTATION
5-JAN-87 11:20 ATHENA-MOD 1.2A ON HOST AP3000
HOLDUP FROM 0.100 TO 5.000 M. IN BRANCH HTCH